IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	: BRONICKI, Youval et al.
Serial No.	: 10/675,915
Filed	: October 1, 2003
For	: A METHOD, A LANGUAGE AND A SYSTEM FOR THE DEFINITION AND IMPLEMENTATION OF SOFTWARE SOLUTIONS

Group Art Unit 2193
Examiner: WOOD, William H.

Tel Aviv, Israel
August 23, 2007

Hon. Commissioner of Patents and Trademarks
Alexandria, VA

Sir:

DECLARATION UNDER 37 CFR SEC 1.132

I, the undersigned, Ofer Brandes, of Kfar-Saba, Israel hereby declare as follows:

Background Information

1. I am a co-inventor of the subject case.

2. I am currently employed as the CEO of Tersus Software Ltd., and I also hold the position of Chief Technology Officer (CTO) of Carmel Ventures, both in Herzliya, Israel.

3. I have previously been employed by the following Israel firms (partial list):

a) SunGard Business Integration as CTO (6 years)
b) Tovna Translation Machines Ltd. as Chief Programmer (10 years)

4. My educational qualifications include:
B.A in Mathematics and Computer Science, cum laude (1987, The Hebrew University of Jerusalem)

OB.

Overview Statements

5. The explanations contained in the following paragraphs address the uniqueness of the present invention over prior art solutions. The explanations do not replace the detailed background and technical details that were already provided in the patent application. These explanations are brought here to challenge the relevance of the Examiner's rejection statements, as stated in the Office Action (OA).

6. The present invention comprises a combination of three major elements:

1) a visualizable computer executable modeling language for the definition of software solutions;
2) a modeling environment for visually defining the software solutions in the modeling language; and
3) a runtime engine that executes solutions defined in the modeling language.

7. The Examiner has rejected claims 1-12 and 14-16 under 35 U.S.C. 102(b) as being anticipated by Williams (US Patent No. 5,850,548).

My comments

8. Williams can be summarized as disclosing a visual programming environment based on a high level hierarchical data flow model. Like the present invention, it discloses a dataflow language using data flow diagrams.

9. Dataflow languages using data flow diagrams, developed since the 1960's, define applications in terms of the data flowing between operation components. Operations components have inputs and outputs, and a component is executed as soon as all of its inputs are available.

10. In dataflow languages operations consist of "black boxes" with inputs and outputs, all of which are always explicitly defined. They run as soon as all of their inputs become valid, as opposed to when the program encounters them. Whereas a traditional program essentially consists of a series of statements saying "do this, now do this", a dataflow program is more like a series of workers on an assembly line, who will do their assigned task as soon as the materials arrive. This is why dataflow languages are inherently parallel; the operations have no hidden state to keep track of, and the operations are all "ready" at the same time. A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system.

O.B.

11. The most well-known dataflow language using data flow diagrams is LabView. The present invention, as well as Williams, share many properties of LabView and similar languages, most notably:

Execution is determined by the structure of a graphical diagram in which the developer connects processes by drawing data flow links;

The data flow links propagate data between processes;

A process can execute as soon as all its input data is available;

The data flow defined by linking inputs and outputs of processes completely defines the execution sequence;

The type of data flowing through a data flow link is defined by the type of its source.

The data flow links automatically adjust to various types of data; and

The graphical approach allows non-programmers to build software applications by simply dragging and dropping virtual representations of library components.

12. Although both Williams and the present invention are based on common dataflow notions and constructs, the patentable parts of each are the specific characteristics of each language and the development process defined for building software applications with it.

13. Williams is a step forward in relation to previous dataflow languages, but is not a completely programming-less application development. Williams describes a specific dataflow language, wherein a user constructs a program by selecting one or more components from a library, which displays components in a tabbed palette. Using a visual editor of the system, the user may drill-down into the internals of a component for modifying its logic. The user connects various components via the component "ports", which allow access to properties of the component. Components of the system may be nested within other components to an arbitrary level. Thus, high-level component manipulations are performed "visually," as illustrated in Figs 1-9B.

14. However, like other dataflow languages but unlike the present invention, Williams cannot model arbitrarily complex data structures and cannot specify optional inputs, which together can give developers significant additional expressive power, equivalent to the expressive power of the leading procedural and object oriented language.

15. The following list contains some of the major differences between the present invention and Williams:

O.B.

a) In claim 1 of Williams, a "program component," corresponding to the "process model" of the present invention, has properties, some of which "surface" as "port connections," corresponding to the "slots" of the present invention. In the present invention a slot is not a property of a process model as in Williams, but rather a sub-component of a process model.

b) In Williams, connecting ports graphically is a mandatory step (step (e) in claim 1). By contrast, in the present invention "flow rules" are optional components.

c) In Williams, the user needs to "surface" a property as a "port connection" to make it available for connection (see step (e)(i) of claim 1). In the present invention there is no "surfacing" step.

16. The present invention includes several notions and constructs which do not exist in Williams:

Claim 1: "Data models" (atomic and composite), which can be sub-components of process models or other data models. Williams only describes properties of components.

Claim 1 has been amended by incorporating a portion of the language of claim 5. The present invention system classifies input slots as mandatory or optional. This, when combined with the ability to model arbitrarily complex data structures, gives a significant expressive power, which is lacking in other dataflow languages.

Claim 7: Mapping between database tables and data models.

Claim 10: Various composition methods of composite data models (concatenation, collection, selection, recurrence).

17. I believe that the Examiner's rejection of claim 1 is overcome by the amendments in the response accompanying this affidavit. In addition, since claims 2-12 and 14-16 are dependent on claim 1, the rejection is overcome without further amendments.

18. I believe that the use of the Williams reference to form the basis of the Sec. 102(b) rejection is improper, and respectfully request that it be withdrawn as well.

19. The Examiner has rejected claim 13 under 35 U.S.C. 103(a) as being unpatentable over Williams. In view of the amendment to claim 1, as above, and since claim 13 is indirectly dependent on claim 1, it is believed that the rejection is overcome without further amendment.

O. B.

20. The Examiner has rejected claims 17 - 20, 24 - 25 and 27 - 28 under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Parr et al, (US20020095653).

My comments

21. Parr discloses a visual architecture software language to define discrete software modules and data flow and control flow between them. It includes a code generation step (called "compilation") to convert the visual representation into executable code in the form of executable instructions and data tables.

22. Code generation is a known technique to convert the definition of a software component (e.g. a software application) from one representation to another, typically from a high-level, user-friendly notation to a lower-level notation.

23. Although both Parr and the present invention use code generation, the patentable parts of each are the specific characteristics of each language and the code generation algorithm used to generate code from it.

The code generator of the present invention is inherently related to the visual modeling language of the present invention, and it has to create code that is functionally 100% equivalent to the operation performed by the runtime engine described in claim 17 of the present invention. None of the features of this complex code generator can be inferred, logically or practically, from the mere notion of "compilation" mentioned by Parr, which discusses code generation in very general terms, without explaining what "compilation" means (note that Parr's claim 16 contains an unfinished sentence that says nothing on the generated code: "...compiling the visual representation to generate executable code. said executable code comprising").

24. The Examiner has rejected claim 21 - 23 under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Goodwin et al, (US 6,199,195).

My comments

25. Goodwin deals with "automatically generated object-oriented source code." This is not the case with the code generator of the present invention.

26. Furthermore, the essence of the present invention is the completeness and preciseness of the language, which enable the needed code generation.

Although both Goodwin and the present invention use code generation, the patentable parts of each are the specific

O.B.

characteristics of each language and the code generation algorithm used to generate code from it.

The code generator of the present invention is inherently related to the visual modeling language of the present invention, and it has to create code that is functionally 100% equivalent to the operation performed by the runtime engine described in claim 17 of the present invention.

27. The Examiner has rejected claims 26 and 29 under 35 U.S.C. 103(a) as being unpatentable over Williams, in view of Parr et al, further in view of Goodwin et al, (USPN 6,199,195).

28. I believe that there is no reason to consider the combination of prior art references to Williams, Parr and Goodwin as making the present invention unpatentable, since they include no provision for totally eliminating the need for writing code in any programming language to implement software applications.

29. In summary, the following conclusions may be drawn:

a) Williams is a step forward, but is not a completely programming-less application development;

b) Williams cannot model complex data structures and cannot specify optional inputs;

c) The present invention system enables the modeling of arbitrarily complex data structures and classifies input slots as mandatory or optional. This gives a significant expressive power, which is lacking in other dataflow languages;

d) Although Parr, Goodwin and the present invention use code generation, the patentable parts of each are the specific characteristics of each language and the code generation algorithm used to generate code from it.

30. This declaration is given in support of the patent prosecution efforts in the present application, before the USPTO.

31. I declare that all the statements made herein of my own knowledge are true, and that all statements made on information and knowledge are believed to be true, and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Sec. 1001 of Title 18 of the United States Code, and the willful false statements may jeopardize the validity of the application and any patent issuing thereon.

Signed this 23rd day of August 2007.

Ofer Brandes

Ofer Brandes